PENDING CLAIMS

1. (Twice Amended) A method for processing software instructions comprising:

decomposing a macroinstruction into a plurality of microinstructions,

issuing all of the plurality of microinstructions simultaneously, in parallel,

executing all of the plurality of microinstructions simultaneously, in lockstep using functional units in a floating point unit,

determining whether an exception occurs in any of the microinstructions, before writing results of the executing to result registers, and

if an exception occurs in any of the microinstructions, canceling all of the microinstructions and preventing the results of the executing from being written to the result registers, and

if no exception occurs in any of the microinstructions, writing the results of the executing to the result registers.

3. (Amended) The method of claim 1, wherein the microinstructions are executed on separate execution units, but appear as though they were executed on a single execution unit.

4. (Amended) The method of claim 1, wherein all of the microinstructions are executed on the same clock cycle.

5. (Amended) The method of claim 1, wherein the microinstructions are executed over multiple clock cycles.

6. The method of claim 1, wherein the method is implemented in a system emulating SSE instructions.

7. The method of claim 6, wherein the system allows a single instruction to operate on multiple single-precision ("SP") floating-point ("FP") values.

8. (Amended) The method of claim 1, further comprising updating a flag based upon a result of the execution of the microinstructions.

9. (Amended) The method of claim 1, further comprising,

if an unmasked exception occurs, canceling the execution of all of the plurality of microinstructions, without regard to the relative ages of each of the plurality of microinstructions, and invoking a microcode handler, and

if an unmasked exception does not occur, updating at least one exception flag by independently generating a logical OR of exceptions for a plurality of functional units.

10. (Amended) A method for processing software instructions comprising:

providing two microinstructions to emulate a high-half and a low-half SSE operation,

forcing the high-half and low-half operations to issue in parallel,

dispatching the high-half and low-half operations simultaneously to a first FP unit and to a second FP unit, respectively,

executing the high-half and low-half operations simultaneously, in lockstep,

generating a signal from an emulator's hardware,

sending the signal to the first and second FP functional units,

determining whether an exception is taken in either the first or the second FP unit,

if an exception is taken in either the first or second FP unit,

preventing results from the high-half and low-half operations from being written to result registers, and

canceling both the high-half and low-half operations, and

updating MXCSR flags based upon the results of the first and second FP units.

11. The method of claim 10, wherein the flushing of a result in the other FP unit does not depend upon the relative ages of the two microinstructions.

12. (Twice Amended) A computer system comprising:

a processor comprising,

a floating point unit comprising a plurality of functional units adapted to execute microinstructions;

a ROM;

a plurality of floating point registers;

wherein the processor is configured to emulate an instruction set by:

decomposing a macroinstruction into a plurality of microinstructions;

issuing all of the plurality of microinstructions simultaneously, in parallel, to the functional units,

determining whether an exception occurs in any of the functional units,

setting result registers for results of each of the functional units only if no exception occurs in any of the functional units, and

if an exception occurs in any of the microinstructions, canceling all of the microinstructions and preventing the setting of result registers for all of the functional units.

13. (Amended) The computer system of claim 12, wherein the processor is further configured to emulate the instruction set by executing all of the microinstructions.

14. (Amended) The computer system of claim 13, wherein the microinstructions are executed on separate execution units, but appear as though they were executed on a single execution unit.

15. (Amended) The computer system of claim 14, wherein the processor is further configured to emulate an instruction set by updating a flag based upon a result of the execution of the microinstructions.

17. The computer system of claim 12, wherein the instruction set is a SSE instruction set.

18. The computer system of claim 17, further comprising an FP register having 82 bits, wherein the computer system uses two FP registers to emulate four 32-bit single-precision, floating point values in an SSE register.

19. The method of claim 1, wherein the step of issuing comprises forcing the microinstructions to issue simultaneously, in lockstep with each other, and wherein the step of canceling comprises canceling all of the plurality of microinstructions without regard to the relative ages of the microinstructions and without using a backoff mechanism.

20. The method of claim 10, wherein the step of forcing the high-half and low-half operations to issue in parallel comprises causing the high-half and low-half operations to execute simultaneously in lockstep with each other, and wherein the step of flushing a result comprises canceling each of the high-half and low-half operations if an exception is taken in either the first or second FP unit.

21. The method of claim 1,

wherein the step of executing comprises executing using a plurality of functional units of a floating point unit,

further comprising:

generating a signal using emulation hardware, wherein the signal indicates that the functional units are emulating an SSE instruction; and

sending the signal to the functional units, and

wherein the step of determining comprises determining after the signal is sent.

22. The system of claim 12, wherein the processor is further configured to emulate an instruction set by:

generating a signal using emulation hardware, wherein the signal indicates that the functional units are emulating an SSE instruction; and

sending the signal to the functional units, and

wherein the step of determining comprises determining after the signal is sent.